Технически Университет – София



**КУРСОВ ПРОЕКТ**

по дисциплина

**„Компютърна графика“**

на тема:

**Tessellation Shaders**

**Изготвил:**  **Проверил:** Дияна Антовска, ф.No123216012 ас. Десислав Андреев

4 курс, 9 поток, 47 група

ФКСТ, КСИ, ПС

София 2019

**Въведение**

1**Shader** в компютърната графика представлява вид на копмютърна програма която предимно се ползва за **shading** в 3D сцени - създаването на съответни нива на светлина, тъмнина и цветове в едно рендерирано изображение, т.е. определяне на цвета на точки чрез интерполиране между точките с известна осветеност. За тази цел извършва различни спциализирани функции които са част от специални ефекти на компютърната графика, пост-обработка на видеа несвързани със създаването на сенки или пак, извършва функции които изобщо не са свързани с графиката.

Трацидиционалните shader-и изчисляват ефектите на рендерите на графичния хардуер с висок степен на гъвкавост. Повечето shader-и са писани за и работят на GPU-то. С развитието на видео картите, голям брой на библиотеки като OpenGL, Direct3D започнаха да поддържат shader-и. Съществуват 3 вида shader-и: 2D, 3D и Compute Shaders.

**2D shader-ите** се прилагат на изображения, т.н. текстури в компютърната графика. Те модифицират атрибути на пикселите. Също могат да участват в рендерирането на 3D геометрията. Единствения 2D shader е **Pixel Shader**-a. Pixel Shader-ите изчисляват цвета и други атрибути на всеки фрагмент – рендерираща единица която влияе на един пиксел.

**3D shader-ите** се прилагат на 3D модели или геометрични обекти, но могат да достъпват цветовете и текстурите които се ползват за изчертаване на модела или **mesh**-a (колекция от възли, рабове и повърхнини които дефинират обект). **Vertex Shader** – най-често срещаните shader-и, целта на които е да трансформират 3D позицията на всеки възел в 2D координати във виртуалното пространство на екрана (също и стойността на дълбочината за z-buffer-a). Те влияат върху позиция, цвят, координати на текстурите но не могат да създават нови възли. **Geometry Shader** – генерира примитиви като точки, линии, триъгълници при подадени друг вид на примитиви. **Primitive Shader-ите** са близки с **Compute Shader-ите** (добавят допълнителни ефекти в алгоритмите за анимация и осветление) само че имат досъп до нужни данни за процесиране на геометрията. **Tessellation Shaders** – последния вид 3D shader-и и тяхната същност, функция и приложения са концентрацията в този реферат.

**Tessellation Shader – Същност, функция**

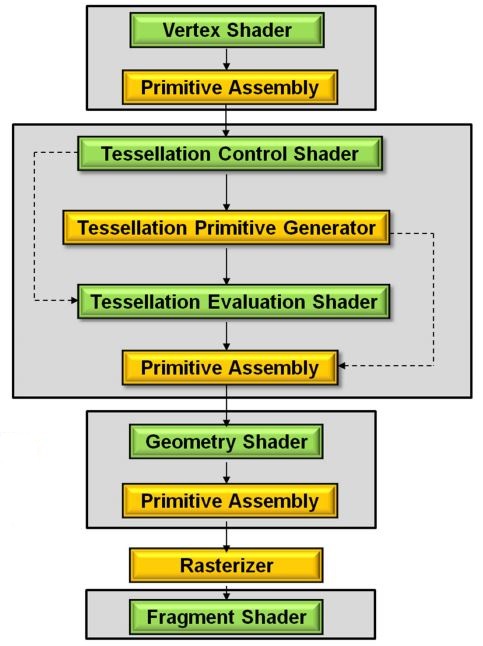
2Tessellation е процеса който разделя една повърност на гладки тръгълни mesh-ове. Tessellation Shader-ите интерполират геометрия (точки, линии, полигони) с цел създаване на допълнителна геометрия която ще направи адаптивно подразделение в зависимост от размер или кривини (curves). Общо взето, тяхната цел е да увеличат качеството с изобразяване на релефа на геометричните модели в изходното изображение.

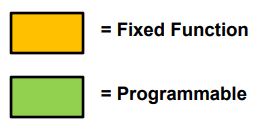
Geometry Shader-ите също я имат способността за генериране на геометрия от вече съществуваща. Разликата е че те могат да създадат нова, различна топология, докато Tessellation Shader-ите добавят такива елементи които са същи като началните. Примерно, от един сегмент могат да се получат повече линии, или от един триъгълник повече триъгълници.

3Tessellation Shader-ите се използват за следното:

* Адаптивно подразделение в зависимост от различни критерии (размер, криви, и т.н.)
* Могат да се предостават по-груби модели, но да се показат по-фини – geometric compression
* Може да се адаптира визуално качество до необходимото ниво на детайлност
* Могат да се представат по-гладки силуети

Tessellation Shader-а се прилага върху изображението след Vertex Shader-а и преди Geometry Shader-а.

****



Tessellation Shader-а се съдържа от Tessellation Control Shader (TCS), Tessellation Primitive (Pattern) Generator (TPG) и Tessellation Evaluation Shader (TES). TCS и TES са програмируеми, докато TPG е фиксирана функция. За да се активира Tessellation Shader-а, задължително е преди това Vertex Shader-a да се изпълни за всеки възел който е посочен от командата за изчертаване.

**Tessellation Control Shader (TCS)** – чете входа който Vertex Shader-a му подава, който вход всъщност представлява **patch** (примитива, която се състои от n на брой възли). 4Главната работа която трябва да се свърши на това ниво е да се определи нивото на tessellation което трябва да се приложи на дадена примитива и да се извърши нужната трансформация на входните данни. Тези изчисления се правят в зависимост от разстоянието до окото, обхвата на екрана, кривите на обектите, грапавостта на изместване.

TCS може да промени големината на примитивата, т.е. да добави повече или да премахне някои от възлите, но не може да отстрани самата примитива или да създаде нова. TCS е опционален, т.е. ако няма нужда от никаква трансформация, Vertex Shader-a ще изпрати данните директно на TPG.

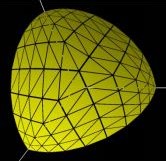
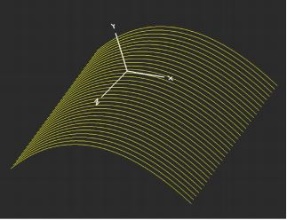
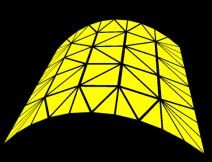
5Има едно извикване за един изходен възел. Примерно, ако командата за изчертаване изчертае 20 примитиви и всяка изходна примитива съдържа 4 възли, TCS ще се извика 80 пъти – 80 x, y, z координати за всеки възел.

**Tessellation Primitive Generator (TPG)** – Извиква се веднъж за всяка примитива – теселира кривата или повърхността в **u, v, w** координати, само ако TES е активен в текущата програма. Това е фиксирана функция. Като вход ги приема възлите от TCS и създава нови възли за триъгълниците, четириъгълници или изолиниите. За триъгълниците са са потребни u, v и w, докато за останалите две само u и v.

На TPG му влияат следващите фактори: нивата на tessellation, които зависат от TCS или тези по подразбиране, разстоянието между възлите, входните примитиви и реда по който ще се генерират.

Tessellation Shader-ите могат да произведът следващите вградени шаблони:

Triangle pattern Quad Pattern Isoline Pattern

****

6**Tessellation Evaluation Shader (TES)** – На входа му се подават абстрактните координати генерирани от TPG, заедно с координатите на възлите генерирани от TCS. Всичките тези координати ги използва за генериране на крайните стойности за възлите. TES е необходим за създаване на tessellation – ако няма TES, процеса на tessellation изобщо не се случва. Колко пъти ще се извика TES зависи от различните имплементации, но се извиква най-малко пъти колкото има абстрактни координати генерирани от TPG.

**Изводи**

**Източници**

1. <https://en.wikipedia.org/wiki/Shader>
2. <https://www.haroldserrano.com/blog/what-is-tessellation-in-opengl>
3. <http://web.engr.oregonstate.edu/~mjb/cs519/Handouts/tessellation.1pp.pdf>
4. <https://www.khronos.org/opengl/wiki/Tessellation>
5. <https://www.khronos.org/opengl/wiki/Tessellation_Control_Shader>
6. <https://www.khronos.org/opengl/wiki/Tessellation_Evaluation_Shader>